"Term Partition" for Mathematical Induction

Pascal Urso, Emmanuel Kounalis

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

"Term Partition" for Mathematical Induction

Urso Pascal — Kounalis Emmanuel

N° 4565

Septembre 2002

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4565--FR+ENG

INRIA
SOPHIA ANTIPOLIS

"Term Partition" for Mathematical Induction

Urso Pascal , Kounalis Emmanuel

Thème 2 — Génie logiciel
et calcul symbolique
Projet COPRIN

Rapport de recherche n° 4565 — Septembre 2002 — 19 pages

Abstract: A key new concept, *term partition*, allows to design a new method for proving theorems whose proof usually requires mathematical induction. A term partition of a term t is a well-defined splitting of t into a pair (a, b) of terms that describes the *language of normal forms of the ground instances of* t.

If $\mathcal{A}$ is a *monomorphic* set of axioms (rules) and (a, b) is a term partition of t, then the normal form (obtained by using $\mathcal{A}$) of any ground instance of t can be "divided" into the normal forms (obtained by using $\mathcal{A}$) of the corresponding ground instances of a and b. Given a conjecture $t = s$ to be checked for inductive validity in a theory $\mathcal{A}$, a partition (a, b) of t and a partition (c, d) of s is computed. If $a = c$ and $b = d$, then $t = s$ is an inductive theorem of $\mathcal{A}$.

The method is conceptually different to the classical theorem proving approaches since it tries to directly mechanize the ω-rule. It allows to obtain elegant and natural proofs of a large number of conjectures (including non-linear ones) without additional lemmas and/or generalizations.

Key-words: Mathematical Induction, Automated Reasoning, Equational Reasoning, Rewriting

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Induction mathématique par « Partition de termes »

Résumé : Un concept clé, *la partition de termes*, permet de définir une nouvelle méthode pour prouver des théorèmes dont la preuve est habituellement fournie par induction mathématique. La partition d'un terme t est constitué d'un découpage bien-défini de t en un couple de termes (a, b) qui décrit le *langage des formes normales des instances closes de* t.

Si $\mathcal{A}$ est un ensemble d'axiomes *monomorphique* et si (a, b) est une partition du terme t, alors la forme normale (selon $\mathcal{A}$) de toute instance close de t peut être « découpée » selon les formes normales (selon $\mathcal{A}$) des instances closes correspondantes de a et b. Étant donnée une conjecture $t = s$ dont la validité inductive doit être vérifiée pour la théorie de $\mathcal{A}$, une partition (a, b) de t et une partition (c, d) de s est calculée. Si $a = c$ et $b = d$, alors $t = s$ est un théorème inductif pour $\mathcal{A}$.

Cette méthode est conceptuellement différente des approches classiques de la preuve de théorèmes car elle essaye d'automatiser directement la règle-ω. Cela nous permet d'obtenir des preuves élégantes et naturelles d'un grand nombre de conjecture sans l'apport de lemmes et/ou de généralisations supplémentaires.

Mots-clés : Induction mathématique, Raisonnement automatique, Logique équationnelle, Réécriture

1 Introduction

The need to be able to prove theorems by induction appears in many applications including number theory, program verification, and program synthesis. We assume familiarity with the basic notion of equational logic and rewrite systems (see for instance [8]).

A many-sorted signature Σ is a pair $(\mathcal{S}, \mathcal{F})$ where $\mathcal{S}$ is a set of *sorts* and $\mathcal{F}$ is a finite vocabulary of *functions symbols*. Let $T(\mathcal{F}, \mathcal{X})$ denote the set of well-sorted *terms* built out of function symbols taken from $\mathcal{F}$ and a denumerable set $\mathcal{X}$ of free-sorted *variables*. We assume that $\mathcal{F}$ contains at least one constant symbol by sort. Thus, the set $T(\mathcal{F})$ of *ground* terms (variable-free), is non-empty. If t is a term and θ is a (ground) substitution of (ground) term for variable in t, then $t\theta$ is a *(ground) instance* of t. Finally, an *equation* e is an element of $T(\mathcal{F}, \mathcal{X}) \times T(\mathcal{F}, \mathcal{X})$ and is written as $t = s$.

An equation $t = s$ is a *deductive consequence* of a set $\mathcal{A}$ of equations if it is valid in any model of $\mathcal{A}$. It is well know that $t = s$ is a deductive consequence of $\mathcal{A}$ if and only if $t =_{\mathcal{A}} s$. Here, $=_{\mathcal{A}}$ denotes the smallest monotonic congruence that contains $\mathcal{A}$. An equation is an *inductive consequence* of a set $\mathcal{A}$ of equations if it is valid in the initial (standard) model. In proof theoretical terms, an equation $t = s$ is said to be an ***inductive theorem***, denoted $t =_{ind(\mathcal{A})} s$, if and only if $t\sigma =_A s\sigma$ for all ground instance of $t\sigma = s\sigma$ of $t = s$. Thus, the proof of $t = s$ depends on the proof of infinite number of ground instances of $t = s$.

To establish inductive consequences, classical theorem proving provides either explicit induction [4, 15] or implicit induction [1, 2, 3, 5, 6, 7, 10]. However, inductive proofs often diverge. The hardest problem in using either approach is to find the appropriate *induction schemes* as well as the suitable *lemmas* needed for the proof. As a "simple" example, consider the theorem:

$$x * (x + (x * x)) = (x * x) + (x * (x * x)) \tag{1}$$

Addition $(+)$, and multiplication $(*)$ are defined by means of the equations, $\mathcal{A} = \{x + 0 = x;\ x + s(y) = s(x + y);\ x * 0 = 0;\ x * s(y) = x * y + x\}$, where $s(x)$ represents the successor of x (i.e $x + 1$).

The proof attempt begins with a simple induction step on x. The proof of basis case $0*(0+(0*0)) = (0*0)+(0*(0*0))$ is trivial. In the step case, the induction hypothesis is (1) and the induction conclusion is $s(x)*(s(x)+(s(x)*s(x))) = (s(x)*s(x))+(s(x)*(s(x)*s(x)))$. Simplifying the induction conclusion with the definitions above gives,

$$s(s(x) * (s(x) + ((s(x) * x) + x)) + x) = s(s((s(x) * x) + x) + (s(x) * ((s(x) * x) + x) + x))$$

The induction hypothesis cannot be used to simplify further so another induction is performed. Unfortunately this generates a diverging sequences of subgoals. The proof of (1) simply fails. The problem is that the prover repeatedly tries an induction on x but is unable to simplify the successor functions that this introduces on the first positions of $+$ and $*$. *This failure especially tied to the classical induction setting which is based on the explicit or implicit use of induction hypothesis.* Equation (1) cannot be proven when given just the

above definitions without a suitable *generalization* of induction hypothesis[1]. For instance, to prove this equation the user has to provide the lemma $x * (y + z) = x * y + x * z$.

A *rewrite system* $\mathcal{R}$ is a set of oriented equations $\{l \rightarrow r\}$, called *rewrite rules*. A rule is applied to a term t by finding a subterm s of t that is an instance of the left side l (i.e., $s = l\sigma$) and replacing s with the corresponding instance $(r\sigma)$ of the rule's right side. One computes with $\mathcal{R}$ by repeatedly applying rules to rewrite an input term until a *normal form* (unrewritable term) is obtained. In case where $\mathcal{A}$ can be compiled into a ground convergent $\mathcal{R}$ rewrite system, for a ground substitution σ, we can decide $t\sigma =_{\mathcal{A}} s\sigma$ by testing for syntaxic identity the $\mathcal{R}$-normal forms of $t\sigma$ and $s\sigma$ (i.e. Is $nf(t\sigma) \equiv nf(s\sigma)$? where $nf(t\sigma)$ denotes the $\mathcal{R}$-normal form of $t\sigma$).

In terms of these definitions, a deep analysis of the normal forms of the ground instances of the both sides of (1) shows the existence of two terms $a = x * x$ and $b = x * (x * x)$ that verify the following property: for all ground substitutions θ,

$$nf((x * (x + (x * x)))\theta) = nf((x * x)\theta) \otimes nf((x * (x * x))\theta)$$
$$\text{and } nf(((x * x) + (x * (x * x)))\theta) = nf((x * x)\theta) \otimes nf((x * (x * x))\theta)$$

Roughly speaking, the normal form of a ground instance of $x * (x + (x * x))$ is obtained by replacing the innermost symbol (i.e. 0) of the normal form of the ground instance of $x * x$ by the normal form of the ground instance of $x * (x * x)$ (the semantics of $\otimes$). Similarly, the normal form of a ground instance of $(x * x) + (x * (x * x))$ is obtained by replacing the innermost symbol (i.e. 0) of the normal form of the ground instance of $x * x$ by the normal form of the ground instance of $x * (x * x)$. Since both sides of (1) can be divided into terms $a = x * x$ and $b = x * (x * x)$ (a term partition, see definition 5) we may conclude that (1) is an inductive theorem (see theorem 1).

Our *approach tries to capture this intuition as to why the above equation is an inductive theorem.* The central part of this paper is to show how our method enables to obtain theorems for automated induction provers, completely automatically from the function definitions alone.

The organization of this paper is as follows. In Sect. 2, the class of monomorphic rewrite system is provided. In Sect. 3, an outline of our approach with a "simple" example is presented. Section 4 introduces the key concept of term partition and shows how to use it to prove inductive theorems. Section 5 describes how to compute a term partition. In Sect. 6 we present our general inductive procedure and applies it to some *hard* examples[2]. Appendix proves formally all our results.

[1] Please note that the most advanced heuristic-based methods ([14, 9, 11, 13]) fail also to produce the necessary lemmas and/or generalizations of the induction hypothesis for the proof to go through for every example of the present paper.

[2] A hard theorem requires either a change of induction rule, or additional lemmas and generalizations for the proof to go through.

2 Monomorphic Rewrite Systems

A *ground convergent* rewrite system $\mathcal{R}$ over a set of function symbol $\mathcal{F}_\mathcal{R}$ is *terminating* – w.r.t. a reduction ordering $\succ$ – and *ground confluent* – i.e. $\mathcal{R}$ has the Church-Rosser property on ground terms. Termination implies that there is at most one $\mathcal{R}$-normal form for any term. We assume that $\mathcal{F}_\mathcal{R}$ can be partitioned into *free constructors* $\mathcal{C}_\mathcal{R}$ and *defined symbols* $\mathcal{D}_\mathcal{R}$, such that every ground term with a defined symbol can be made equal (using $\mathcal{R}$) to a ground term built upon constructors only. We shall from now on regard our set $\mathcal{A}$ of equations as a rewrite system $\mathcal{R}$.

The set $TS(\mathcal{R}) = \{t \mid t$ is constant or t is of the form $c(x_1, x_2, \ldots, x_n)$ with $c \in \mathcal{C}_\mathcal{R}$ and $x_1, \ldots, x_n) \in \mathcal{X}\}$ is a *test set* for $\mathcal{R}$ [12]. A *test set substitution* σ is a substitution with values in $TS(\mathcal{R})$. If t is a term $nf(t)$ denotes the normal form of term t using $\mathcal{R}$. Let $(t = s)^\prec$ denote the instances of equation $t = s$ that are smaller (w.r.t. $\succ\succ$ the multiset extension of $\succ$) than $t = s$.

The following function GEN encompasses the classical definition of induction.

Definition 1. $\mathbf{GEN(t, s)} = \{nf(t\sigma) = nf(s\sigma) \mid \sigma$ *is test set substitution,* $nf(t\sigma) \neq nf(s\sigma)$*, and* $nf(t\sigma)$ *(resp.* $nf(s\sigma)$*) is computing using* $T = \mathcal{R} \cup \{(t = s)^\prec\}$.

If t is a term, $type(t)$ denotes the sort of t and $dom(t)$ is the set of positions of t. ε denotes the *empty position*, p, q, ... denote *positions* in a term. $|p|$ is the length of position p and $<$ is the *prefix ordering*. For notational convention, we shall represent positions as lists of digits from $\{1 \ldots 9\}$. t/p denotes the *subterm of* t *at position* p. We write $t[s]_p$ the result of replacing the subterm of t at position p by s. Finally $t(p)$ denotes the *symbol of* t *at position* p.

The **monomorphic rewrite systems** are a subset of ground convergent rewrite systems. Their main interest is that their ground normalized terms can be viewed as "lists". Now, *in a list we are able to identify a "head" and a "tail" part and then to split a ground normalized term.*

Definition 2. *A ground convergent rewrite system* $\mathcal{R}$ *is* monomorphic *if there is only one constant by sort* T *– denoted* $\bot_T$ *– and all ground term* t *in* $\mathcal{R}$*-normal form satisfies* $\forall (u, v) \in dom(t)$ *such that* $sort(t/u) = sort(t/v) = sort(t)$*, we get* $u \leq v$ *or* $v \leq u$.

Example 1. Let $\mathcal{R}$ be the following system:

$$
\begin{array}{llll}
\{\ x + s(y) \to s(x + y); & x + 0 \to x; & e(x, s(y)) \to e(x, y) * x; & e(x, 0) \to s(0); \\
x * s(y) \to (x * y) + x; & x * 0 \to 0; & m(x, s(y)) \to x + m(x, y); & m(x, 0) \to 0; \\
\Sigma(s(x)) \to s(x) + \Sigma(x); & \Sigma(0) \to 0; & \Sigma I(s(x), y) \to \Sigma I(x, s(y + x)); & \Sigma I(0, y) \to y; \\
r(c.l) \to ap(r(l), c.\varnothing); & r(\varnothing) \to \varnothing; & ap(c.l, L) \to c.ap(l, L); & ap(\varnothing, L) \to L; \\
R(c.l, L) \to R(l, c.L); & R(\varnothing, L) \to & L\ \} &
\end{array}
$$

$\mathcal{R}$ is monomorphic, because the ground terms in $\mathcal{R}$-normal form can be generated by the term grammar:

$$S \to \Xi \mid \Psi, \Xi \to 0 \mid s(\Xi), \Psi \to \varnothing | \Xi.\Psi$$

Further, $TS(R) = \{0, \varnothing, s(x), c.l\}$. Throughout of this paper we shall make free use of rules in $\mathcal{R}$.

Thus, *a rewrite system is monomorphic if its constructors have at most one argument of their own sort*, this argument is called "reflective".

Definition 3. *Let $\mathcal{R}$ be a monomorphic rewrite system and let $t \in T(\mathcal{C}_\mathcal{R}, \mathcal{X})$ be a term; the unique position $p \in dom(t)$ that verifies $|p| = 1$ and $type(t/p) = type(t)$ is called* the reflective argument position *of the symbol that roots t, and is denoted $RA(\mathcal{R}, t(\varepsilon))$. We denote $c[x]$ a* constructor term *such that $c/RA(\mathcal{R}, c(\varepsilon)) = x$.*

For the above rewrite system, we get that $RA(\mathcal{R}, .) = 2$ and $RA(\mathcal{R}, s) = 1$.

Since there is just one reflexive argument position by constructor symbol, in any ground normalized term, the constant of the term's sort is at only one position. Finally, the "join" of two ground normalized terms in a monomorphic theory is defined as follows.

Definition 4. *Let $\mathcal{R}$ be a monomorphic rewrite system and let A and B two ground terms in $\mathcal{R}$-normal form, both of sort T; the* join *of A and B denoted $A \otimes B$ is the term $A[B]_p$ such that $A/p = \bot_T$.*

For instance, $s(0).(0.\varnothing) \otimes s(s(0)).\varnothing = s(0).(0.(s(s(0)).\varnothing))$

3 Outline of our Approach: an Example

The key concept underlying our approach is to split both sides of a given equation $t = s$ into pairs (a, b) and (c, d) and to recursively check whether $a = c$ and $b = d$ (see Theorem 1).

To obtain a pair that describes the head and the tail part of the normal form of the ground instances of a term t we have to identify which particular combination of subterms of t "create" these parts. Further, to find these subterms we have to understand how each function in the rewrite system manages its arguments. Having understood this, we will be able to compute precisely which subterms participate to the building of the head or tail part of the normal form of the ground instances of terms as well as a description of these parts.

To illustrate the essential ideas behind our method let us outline it on a "tricky" example. Assume we wish to prove the non-linear equation $t = s$:

$$r(ap(l, r(l)) = ap(l, r(l)) \tag{2}$$

Our approach to induction consists in computing a term partition (actually a set) for each side of (2) (see induction procedure in Sect. 6). This set is computed in two steps (see Sect. 5).

At the *first* step, the type of the argument positions of the function symbols appearing in $\mathcal{R}$ is characterized. Intuitively, this characterization indicates how each function argument is moved onto another argument position at each step of a rewriting sequence. This type can be precompiled using the definitions 6, 7, and 8 (see Sect. 5).

In our example, position 1 is the upward argument of ap; position 2 is the downward argument of ap; and position 1 is the down-contextual argument of r. Roughly speaking, the *upward* argument position of a function symbol f indicates which subterm in a ground term (with f as root symbol) moves towards the head of the normal form in a rewriting sequence. The *downward* argument position indicates which subterm in a ground term moves down towards the tail of the normal form in a rewriting sequence. The *down-contextual* argument position indicates which subterm in a ground term is used to create a regular series of elements whose normalization leads to the tail of the normal form[3].

At the *second* step, "combinations" of subterms of each side of the equation to split are identified. These combinations are represented by the *top* and *bot* function (see Definition 10) and computed upon a "*top*" (TP) and "*bottom*" (BP) paths (see definition 9).

For instance, we get that $TP(r(ap(l,r(l)))) = 12, TP(ap(l,r(l))) = 1, BP(r(ap(l,r(l)))) = 11$, and $BP(ap(l,r(l))) = 21$. Roughly speaking, a TP (resp. BP) path shows where to look for the subterms that creates the head (resp. the tail) of a normal form of a ground instance of a term. For instance, the head of the normal form of a ground instance of $r(ap(l,r(l)))$ will be the result of computing the normal forms of the ground instance of $r(l)$ (at position 2 for ap) and then applying the definition of r to position 1 (path 12). Similarly, the "rest" of the normal form of a ground instance of $r(ap(l,r(l)))$ can be found by applying the rules of the function symbols that label each position along the path 12. Each path allows us to compute (see definitions 10 and 11) the following sets:

$$DEC(t) = \{(r(ap(l,r(l))),\varnothing);(r(r(l)),r(ap(l,\varnothing)));(\varnothing,r(ap(l,r(l))));(r(r(l)),r(l))\}$$
$$DEC(s) = \{(ap(l,r(l),\varnothing);(l,r(l));(\varnothing,ap(l,r(l)));(ap(l,\varnothing),r(l))\}$$

Each element of these sets constitutes a partition of t and s (see proposition 1). Since $(r(r(l)),\mathbf{r(l)})$ is a partition of $r(ap(l,r(l)))$ and $(l,\mathbf{r(l)})$ is a partition of $ap(l,r(l))$, we may reduce – by "taking off" the tail parts $r(l)$ – the proof of (2) to the proof of $t' = s'$ (see theorem 1):

$$r(r(l)) = l \tag{3}$$

So, we need to check (3) for inductive validity. We iterate the process of computation of a term partition of both sizes of (3) using definition 11. Since a suitable partition for both terms cannot be found using the definition of DEC, the next step of our induction procedure consists of computing the "*patterns*" (see definition 12) of both sides of (3). Roughly speaking, since ground normal forms of terms in monomorphic systems can be viewed as "lists", they can thus be made up of a regular series of similar elements. Intuitively, a pattern is a representation of these elements when they exist.

To compute patterns, two positions $p \in dom(t')$ and $q \in dom(s')$ are first chosen. Then, the normal forms of terms $t'[c.x]_p$ and $s'[c.x]_q$ are computed, where $c.x$ is a non constant element of $TS(R)$ (see Sect. 2). The reason for that is to find a suitable partition for $t'[c.x]_p$

[3]A fourth type of argument position – *up-contextual* – is not present in these terms, it indicates which subterm is used to create a regular series of elements whose normalization leads to the head of the normal form.

that contains $\mathbf{t[x]_p}$. Since $nf(t'[c.x]_{11}) = r(ap(r(l), c.\varnothing))$, we get that $(\mathbf{r(r(x))}, c.\varnothing) \in DEC(nf(t'[c.x]_{11}))$, and thus $pat(t', 11, c.x) = c.\varnothing$. Similarly, since $nf(s'[c.x]_\varepsilon) = c.x$, we get that $(\mathbf{x}, c.\varnothing) \in DEC(s'[c.x]_\varepsilon)$, and thus $pat(s', \varepsilon, c.x) = c.\varnothing$.

Since $pat(t', p, c.x) = pat(s', q, c.x)$, we have to check whether $r(r(\varnothing)) = \varnothing$ is an inductive theorem (see theorem 2). Now, $r(r(\varnothing)) = \varnothing$ trivially holds. Therefore (3) is an inductive theorem. Hence (2) is so.

4 Induction Using "Term Partition"

In this section, we define our approach to mathematical induction that is directly based on the finite specification of the language of the normal forms of the infinitely many ground instances of a term. In other words, our method tries to directly mechanize the ω-rule.

Definition 5. *If $\mathcal{R}$ is a rewrite system and t is a term, then a* partition *of t with respect to $\mathcal{R}$ is the ordered pair (a, b) where a and b are terms such that $var(a) \cup var(b) \subseteq var(t)$ and for every ground substitution θ the following holds: $nf(t\theta) = nf(a\theta) \otimes nf(b\theta)$.*

For instance, if $t = ap(ap(L, r(L)), L)$, then $(L, ap(r(L), L))$ is a partition of t: the normal forms of the ground instances of t are equal to the normal forms of the ground instances of L (at position 1) followed by those of $ap(r(L), L)$. The first variable L is the one at position 121 in t and the second is the one of the position 2 in t. Similarly, $(L, ap(L, L))$ is a partition of $t = ap(L, ap(r(L), L))$. The key property of term partition is the following theorem:

Theorem 1. *Let $\mathcal{R}$ a monomorphic ground convergent rewrite system and $t = s$ be an equation. Assume that (a, b) is a partition of t and (c, d) is a partition of s. If $a =_{ind(\mathcal{R})} c$ then, $t =_{ind(\mathcal{R})} s$ if and only if $b =_{ind(\mathcal{R})} d$. Respectively, if $b =_{ind(\mathcal{R})} d$ then, $t =_{ind(\mathcal{R})} s$ if and only if $a =_{ind(\mathcal{R})} c$.*

Proof. Let θ be a ground substitution. Since (c, d) is a partition of s, $nf(s\theta) = nf(c\theta) \otimes nf(d\theta)$. Since (a, b) is a partition of t, $nf(t\theta) = nf(a\theta) \otimes nf(b\theta)$.

Assume first that $t =_{ind(\mathcal{R})} s$.

Since $\mathcal{R}$ is ground convergent, we get that $nf(t\theta) = nf(s\theta)$. Therefore, we have $nf(a\theta) \otimes nf(b\theta) = nf(c\theta) \otimes nf(d\theta)$. If $a =_{ind(\mathcal{R})} c$ we then have $nf(a\theta) = nf(c\theta)$. So $nf(a\theta) \otimes nf(b\theta) = nf(a\theta) \otimes nf(d\theta)$. By definition 4 we get $nf(b\theta) = nf(d\theta)$ and therefore $b =_{ind(\mathcal{R})} d$. Similarly, $b =_{ind(\mathcal{R})} d$ implies $nf(b\theta) = nf(d\theta)$. So $nf(a\theta) \otimes nf(b\theta) = nf(c\theta) \otimes nf(b\theta)$. By definition 4 we get $nf(a\theta) = nf(c\theta)$ and therefore $a =_{ind(\mathcal{R})} c$.

Assume now $a =_{ind(\mathcal{R})} c$ and $b =_{ind(\mathcal{R})} d$.

Since $\mathcal{R}$ is monomorphic, we get an unique position p such that $nf(a\theta)/p = nf(c\theta)/p = \perp$. We then have, $nf(t\theta) = nf(a\theta) \otimes nf(b\theta) = nf(c\theta) \otimes nf(d\theta) = nf(s\theta)$. □

For instance, since $(L, ap(r(L), L))$ is a common partition of $ap(ap(L, r(L)), L)$ and $ap(L, ap(r(L), L))$, theorem 1 says that $ap(ap(L, r(L)), L) =_{ind(\mathcal{R})} ap(L, ap(r(L), L)$.

5 Computing "Term Partitions"

As seen in Sect. 2, a term partition is computed in two steps. The first step is to understand how the normal forms of the ground instances of a term are built up. The second step uses the first step to extract from a term the combination of subterms that form the partition.

5.0.1 Parsing the Rewrite System. Let $\mathcal{R}$ be a ground convergent rewrite system, and let $\mathcal{R}_f = \{l \to r \in \mathcal{R} \mid l = f(t1, \ldots, tn)$ with $f \in \mathcal{D}_\mathcal{R}$ and $t1, \ldots, tn \in T(\mathcal{C}_\mathcal{R}, \mathcal{X})\}$ be the *definition* of f. Assume that t is a term and θ is a ground constructor substitution. If we were able to identify the subterms in $t\theta$ which could be moved (possibly changed) towards the "top" of the $nf(t\theta)$ or down towards the "bottom" of the $nf(t\theta)$ in any rewriting sequence, then we would be able obtain a term partition.

The type of an argument position allows us to know how each function argument is moved onto another argument position in a rewriting step. Let $\mathcal{R}$ be a monomorphic rewrite system. Let P_f be the *argument position set* of f, define $P_f = \{1, \ldots, ar(f)\})$ where $ar(f)$ is the arity of symbol f. Remember also that $RA(R, f)$ is the reflection argument position of $f \in \mathcal{C}_\mathcal{R}$ (see Sect. 2).

Definition 6. *Let $\mathcal{R}$ be a monomorphic rewrite system, let f be a function symbol in $\mathcal{D}_\mathcal{R}$. $p \in P_f$ is a* downward position*, denoted by $DP(\mathcal{R}, f)$, if for all $l \to r \in \mathcal{R}_f$ there exists just one position q such that $l/p = r/q$ and*
- *Either $q = \varepsilon$,*
- *Or $q = q_1 \ldots q_n$ with for all $i \leq n$, $|q_i| = 1$ and $q_i = RA(\mathcal{R}, r(q_1 \ldots q_{i-1}))$, $q_i = DP(\mathcal{R}, r(q_1 \ldots q_{i-1}))$, or $q_i = p$ with $r(q_1 \ldots q_{i-1}) = f$.*

Example 2. If $\mathcal{R}_R = \{R(c.l, L) \to R(l, c.L);\ R(\varnothing, L) \to L\}$, then by taking $p = 2$ we get $q = 22$ or $q = \varepsilon$. Thus $DP(\mathcal{R}, R) = 2$.

Definition 7. *Let $\mathcal{R}$ be a monomorphic rewrite system, let f be a function symbol in $\mathcal{D}_\mathcal{R}$. $p \in P_f$ is a* upward position*, denoted by $UP(\mathcal{R}, f)$, if for all $l \to r \in \mathcal{R}_f$,*
- *Either $l/p = \perp_{type(l)}$,*
- *Or $l/p = c[x]$ and $r = c[l[x]_p]$.*

Example 3. If $\mathcal{R}_{ap} = \{ap(\varnothing, L) \to L;\ ap(c.l, L) \to c.ap(l, L)\}$, then by taking $p = 1$ we get $l/p = \varnothing$ or $l/p = c.l$ and $r = c.\mathbf{ap}(\mathbf{l}, \mathbf{L})$. Thus $UP(\mathcal{R}, ap) = 1$.

Definition 8. *Let $\mathcal{R}$ be a monomorphic rewrite system, let f be a function symbol in $\mathcal{D}_\mathcal{R}$. For a position $p \in P_f$, if for all $l \to r \in \mathcal{R}_f$*
- *Either $l/p = \perp_{type(l)}$ and $r = \perp_{type(l)}$*
- *Or $l/p = c[x]$ and there exists just one position q s.t. $r/q = l[x]_p$. Then p is*
  - *a* up-contextual position *and denoted by $UCP(\mathcal{R}, f)$ if $q = DP(\mathcal{R}, r(\varepsilon))$,*
  - *a* down-contextual position *and denoted by $DCP(\mathcal{R}, f)$ if $q = UP(\mathcal{R}, r(\varepsilon))$.*

Example 4. If $\mathcal{R}_* = \{x * 0 \to 0;\ x * s(y) \to (x * y) + x\}$, then by taking $p = 2$ we get $l/p = r = 0$ or $l/p = s(y)$ and $r/q = (x * y)$ with $q = DP(\mathcal{R}, +)$. Thus $UCP(\mathcal{R}, ap) = 1$.

Example 5. If $\mathcal{R}_m = \{m(x,0) \to 0;\ m(x,s(y)) \to x + m(x,y)\}$, then by taking $p = 2$ we get $l/p = r = \varnothing$ or $l/p = s(y)$ and $r/q = m(x,y)$ with $q = UP(\mathcal{R},+)$, and thus $DCP(\mathcal{R},ap) = 1$.

Terms with downwards, upward, up-contextual, and traversal argument positions may be divided into two parts as the following lemma shows:

Lemma 1. *Let $\mathcal{R}$ be a monomorphic rewrite system, let t be a term, and let p a position in $P_{t(\varepsilon)}$. For any ground substitution θ, and for any ground terms in $\mathcal{R}$-normal form A and B,*

1. *if $p = DP(\mathcal{R},t(\varepsilon))$, then $nf(t[A]_p\theta) = nf(t[\perp_{type(t)}]_p\theta) \otimes A$*
2. *if $p = UP(\mathcal{R},t(\varepsilon))$, then $nf(t[A]_p\theta) = A \otimes nf(t[\perp_{type(t)}]_p\theta)$*
3. *if $p = UCP(\mathcal{R},t(\varepsilon))$, then $nf(t[A \otimes B]_p\theta) = nf(t[A]_p\theta) \otimes nf(t[B]_p\theta)$*
4. *if $p = DCP(\mathcal{R},t(\varepsilon))$, then $nf(t[A \otimes B]_p\theta) = nf(t[B]_p\theta) \otimes nf(t[A]_p\theta)$*

f	$RA(\mathcal{R},f)$	$DP(\mathcal{R},f)$	$UP(\mathcal{R},f)$	$UCP(\mathcal{R},f)$	$DCP(\mathcal{R},f)$
s	1				
$+$		1	2		
$*$				2	
e					
m					2
Σ					
ΣI		2			
.	2				
ap		2	1		
r					1
R		2			

Table 1. Argument positions for $\mathcal{R}$ (see Example 1).

Partition Sets. At the second step, we have to know how each function symbol argument is moved at each step of rewriting sequence of a ground instance $t\theta$ of a term t. The mechanism is directly based on the argument positions as defined above, since they point out the subterms of t that will constitute the "head" and the "tail" of $nf(t\theta)$.

Definition 9. *Let $\mathcal{R}$ be a monomorphic rewrite system, let t be a term. The* maximum top path *of t, denoted by $TP(t)$, and the* maximum bottom path *of t denoted by $BP(t)$, are computed as*

$$\exists p \in \{RA(\mathcal{R},t(\varepsilon)), UP(\mathcal{R},t(\varepsilon)), UCP(\mathcal{R},t(\varepsilon))\} \Rightarrow TP(t) = pTP(t/p)$$
$$\exists p = DCP(\mathcal{R},t(\varepsilon)) \Rightarrow TP(t) = pBP(t/p)$$
$$\nexists p \in \{RA(\mathcal{R},t(\varepsilon)), UP(\mathcal{R},t(\varepsilon)), UCP(\mathcal{R},t(\varepsilon)), DCP(\mathcal{R},t(\varepsilon))\} \Rightarrow TP(t) = \varepsilon$$

$$\exists p \in \{RA(\mathcal{R}, t(\varepsilon)), DP(\mathcal{R}, t(\varepsilon)), UCP(\mathcal{R}, t(\varepsilon))\} \Rightarrow BP(t) = pBP(t/p)$$
$$\exists p = DCP(\mathcal{R}, t(\varepsilon)) \Rightarrow BP(t) = pTP(t/p)$$
$$\nexists p \in \{RA(\mathcal{R}, t(\varepsilon)), DP(\mathcal{R}, t(\varepsilon)), UCP(\mathcal{R}, t(\varepsilon)), DCP(\mathcal{R}, t(\varepsilon))\} \Rightarrow BP(t) = \varepsilon$$

Example 6. $ap(r(ap(l,n)), R(n,l))$
- $CH(ap(r(ap(l,n)), R(n,l))) = 1CH(r(ap(l,n))) = 11CB(ap(l,n)) = 112CB(n) = 112$
- $CB(ap(r(ap(l,n)), R(n,l))) = 2CB(R(n,l)) = 22CB(l) = 22$

Example 7. $x * m(y, z + x)$
- $CH(x * m(y, z + x)) = 2CH(m(y, z + x)) = 22CB(z + x) = 221CB(z) = 221$
- $CB(x * m(y, z + x)) = 2CB(m(y, z + x)) = 22CH(z + x) = 222CH(x) = 222$

We are now ready to specify the "tops" and the "bottoms" of the normal forms of the ground instances of a term t.

Definition 10. *Let $\mathcal{R}$ be a monomorphic rewrite system, let t be a term, and let p be a top path (resp. a bottom path). The* head part *denoted by $top(t,p)$ (resp. the* tail part *denoted by $bot(t,p)$) are computed as*

$q = RA(\mathcal{R}, t(\varepsilon)) \Rightarrow$	$top(t, q.p) = t[top(t/q, p)]_q$	$bot(t, q.p) = bot(t/q, p)$
$q = DP(\mathcal{R}, t(\varepsilon)) \Rightarrow$		$bot(t, q.p) = bot(t/q, p)$
$q = UP(\mathcal{R}, t(\varepsilon)) \Rightarrow$	$top(t, q.p) = top(t/q, p)$	
$q = UCP(\mathcal{R}, t(\varepsilon)) \Rightarrow$	$top(t, q.p) = t[top(t/q, p)]_q$	$bot(t, q.p) = t[bot(t/q, p)]_q$
$q = DCP(\mathcal{R}, t(\varepsilon)) \Rightarrow$	$top(t, q.p) = t[bot(t/q, p)]_q$	$bot(t, q.p) = t[top(t/q, p)]_q$
	$top(t, \varepsilon) = t$	$bot(t, \varepsilon) = t$

The complement of the head part *denoted by $ntp(t,p)$ (resp. the* complement of the tail part *denoted by $nbt(t,p)$) are computed as*

$q = RA(\mathcal{R}, t(\varepsilon)) \Rightarrow$	$ntp(t, q.p) = ntp(t/q, p)$	$nbt(t, q.p) = t[nbt(t/q, p)]_q$
$q = DP(\mathcal{R}, t(\varepsilon)) \Rightarrow$		$nbt(t, q.p) = t[nbt(t/q, p)]_q$
$q = UP(\mathcal{R}, t(\varepsilon)) \Rightarrow$	$ntp(t, q.p) = t[ntp(t/q, p)]_q$	
$q = UCP(\mathcal{R}, t(\varepsilon)) \Rightarrow$	$ntp(t, q.p) = t[ntp(t/q, p)]_q$	$nbt(t, q.p) = t[nbt(t/q, p)]_q$
$q = DCP(\mathcal{R}, t(\varepsilon)) \Rightarrow$	$ntp(t, q.p) = t[nbt(t/q, p)]_q$	$nbt(t, q.p) = t[ntp(t/q, p)]_q$
	$ntp(t, \varepsilon) = \bot_{type(t)}$	$nbt(t, \varepsilon) = \bot_{type(t)}$

The following definition and proposition *implement* the concept of term partition.

Definition 11. *Let t be a term, then the* partition set *of t, denoted $DEC(t)$, is*

$$\{(top(t,p), nf(ntp(t,p))) \mid p \leq TP(t)\} \cup \{(nf(nbt(t,q)), bot(t,q)) \mid q \leq BP(t)\}$$

Proposition 1. *Every element of $DEC(t)$ is a partition of t.*

Example 8. Consider the term $ap(ap(L,L),L)$ and $p = 11$.
- $top(ap(ap(L,L),L), 11) = top(ap(L,L), 1) = top(L, \varepsilon) = L$
- $nf(ntp(ap(ap(L,L),L), 11)) = nf(ap(ap(\varnothing, L), L)) = ap(L,L))$

Thus, $(L, ap(L,L))$ is a partition of $ap(L, ap(L,L))$.

Example 9. Consider the term $r(ap(r(L'), c.\varnothing))$ and $q = 11$.
– $bot(r(ap(r(L'), c.\varnothing)), 11) = r(top(ap(r(L'), c.\varnothing), 1) = r(top(r(L'), \varepsilon)) = r(r(L'))$
– $nf(nbt(r(ap(r(L'), c.\varnothing), 11)) = nf(r(ap(\varnothing, c.\varnothing))) = c.\varnothing$
Thus, $(c.\varnothing, r(r(L')))$ is a partition of $r(ap(r(L'), c.\varnothing))$.

5.0.2 Patterns. In some cases, Definition 10 does not provide us with a suitable term partition. A reason for this is that $\{(t, \bot), (\bot, t)\}$ may be the only elements of $DEC(t)$. Another reason is that the type of the arguments of some function symbols (e.g. e, ΣI, ...) cannot be specified[4]. The following slight generalization allows to deal with this situation:

Definition 12. *Let $\mathcal{R}$ is a monomorphic rewrite system, t be a term, $c[y] \in TS(R)$, and $p \in dom(t)$. The* pattern *of t, denoted $pat(t, p, c[y])$, with respect to $c[x]$ and p is the term α such that $(\alpha, t[y]_p)$ or $(t[y]_p, \alpha)$ is a partition of $nf(t[c[y]]_p)$.*

Example 10. Consider the term $t = x * (x * x)$. Assume that $s(y) \in TS(R)$. We then have $nf(t[s(y)]_{22}) = x * ((x * y) + x)$. Since, $top(x * ((x * y) + x), 22) = x * x$ and $nf(ntp(x * ((x * y) + x), 22)) = nf(x * ((x * y) + 0)) = x * (x * y)$, we get that $x * x$ is the pattern of $x * (x * x)$ at position 22.

Example 11. Consider the term $t = r(r(l))$. Assume that $c.x \in TS(R)$. We then have $nf(t[c.x]_{11}) = r(ap(r(x), c.\varnothing))$ and $bot(r(ap(r(x), c.\varnothing)), 11) = r(r(x))$, we get that $pat(t, 11, c.l) = nf(nbot(r(ap(r(x), c.\varnothing)), 11)) = nf(r(ap(\varnothing, c.\varnothing))) = c.\varnothing$.

Roughly speaking, a pattern of t provides a partition of the instances of t that obtained by replacing variables in t with non-constant elements of the test set. Since $x * x$ is a pattern of $x * (x * x)$ (example 10), then for all ground substitutions θ such that $x\theta = s(y)\theta$ we have $nf((x * (x * x))\theta) = nf((x * x)\theta) \otimes nf((x * (x * y))\theta)$.

Theorem 2. *Let $\mathcal{R}$ be a monomorphic system and $t = s$ be an equation. Assume that there is a pair $(p, q) \in (dom(t), dom(s))$ such that $t/p =_{ind(\mathcal{R})} s/q$. If for all $c[y] \in TS(R)$ one has $pat(t, p, c[y]) =_{ind(\mathcal{R})} pat(s, q, c[y])$ then, $t =_{ind(\mathcal{R})} s$ if and only if $t[\bot_{type(t)}]_p =_{ind(\mathcal{R})} s[\bot_{type(t)}]_q$. Similarly, if $t[\bot_{type(t)}]_p =_{ind(\mathcal{R})} s[\bot_{type(s)}]_q$ then, $t =_{ind(\mathcal{R})} s$ if and only if for all $c[y] \in TS(R)$ one has $pat(t, p, c[y]) =_{ind(\mathcal{R})} pat(s, q, c[y])$.*

Proof. Let θ be a ground substitution. Then $t/p =_{ind(\mathcal{R})} s/p$ implies $nf((t/p)\theta) = nf((s/q)\theta)$ and for all $c[x] \in TS(\mathcal{R})$, $pat(t, p, c[x]) =_{ind(\mathcal{R})} pat(s, q, c[x])$ implies
$nf(pat(t, p, c[x])\theta) = nf(pat(s, q, c[x])\theta)$
Let A and B be ground terms in $\mathcal{R}$-normal form such that $A = nf((t/p)\theta)$ and $B_{c[y]} = nf(pat(t, p, c[y])\theta)$. Since $\mathcal{R}$ is ground convergent, we get, by substitution property, that $nf(t\theta) = nf(t\theta[nf((t/p)\theta)]_p) = nf(t\theta[A]_p)$. Similarly $nf(s\theta) = nf(s\theta[A]_q)$. We proceed by induction on the structure of A (remember $\mathcal{R}$ is monomorphic)[5].
– Basis case: $t[\bot]_p =_{ind(\mathcal{R})} s[\bot]_q$ implies $nf(t\theta[\bot]_p) = nf(s\theta[\bot]_q)$.

[4]Of course, definition 10 can handle these function if they do not occur at the root of t.
[5]In the proofs, since we deal with only one sort, we note $\bot$ the constant of that sort.

– Induction case: for all A' such that $A = c[A']$ we have that $nf(t\theta) = nf(t\theta[c[A']]_p) = nf(pat(t,p,c[y])\theta) \otimes nf(t\theta[A']_p) = B_{c[y]} \otimes nf(t\theta[A']_p)$. Similarly, $nf(s\theta) = B_{c[y]} \otimes nf(s\theta[A']_q)$, and by induction hypothesis $nf(t\theta[A']_p) = nf(s\theta[A']_q)$. Therefore, $pat(t,p,c[x]) =_{ind(\mathcal{R})} pat(s,q,c[x])$ implies $t =_{ind(\mathcal{R})} s$ if and only if $t[\bot]_p =_{ind(\mathcal{R})} s[\bot]_q$. □

6 Organizing the Inductive Proofs

In this section we show how to organize inductive proofs. Assume we are working with a monomorphic, terminating (w.r.t a reduction ordering $\succ$) and ground confluent rewrite system $\mathcal{R}$ (see Sect. 2). To verify whether $t =_{ind(\mathcal{R})} s$ we run the following procedure:[6]

Function inductive($t = s$) : boolean

If $nf(t) \equiv nf(s)$ **Then Return** TRUE;

If $vars(t) = \varnothing$ or $vars(s) = \varnothing$ **Then Return** FALSE;

For all $(a,b) \in DEC(t)$ **and** all $(c,d) \in DEC(s)$

  If $a \equiv c$ **Then Return** inductive($b = d$);
  If $b \equiv d$ **Then Return** inductive($a = c$);

For all $p \in dom(t)$ **and** all $q \in dom(s)$ **If** $t/p \equiv s/q$

  If $\forall c[x] \in TS(\mathcal{R})$. pat$(t,p,c[x]) \neq$ nul **Then**
    If $\forall c[x] \in TS(\mathcal{R})$. pat$(t,p,c[x]) \equiv$ pat$(s,q,c[x])$ **Then**
      Return inductive$(t[\bot_{type(t)}]_p = s[\bot_{type(t)}]_q)$;
    Else If $nf(t[\bot_{type(t)}]_p) \equiv nf(s[\bot_{type(t)}]_q)$ **Then**
      Return $\forall c[x] \in TS(\mathcal{R})$. inductive(pat$(t,p,c[x]) =$ pat$(s,q,c[x])$);

Return inductive(GEN($t = s$));

Function pat(t, p, c[x]) : term

For all $(a,b) \in DEC(nf(t[c[x]]_p))$

  If $a \equiv t[x]_p$ **Then Return** b;
  Else If $b \equiv t[x]_p$ **Then Return** a;

Return *nul*;

Let us illustrate how our induction procedure succeed in some hard examples:

[6] In an actual implemented algorithm, we can, for more efficiency, compute DEC sets just once for every term or patterns. We can also, not compute patterns for every position in *dom* but just for *inductive positions* as defined in [13].

Example 12. Consider the equation $t = s$: $ap(r(l), ap(l, l)) = ap(ap(r(l), l), l)$
– $(nf(nbt(t, 1)), bot(t, 1)) = (r(l), ap(l, l)) \in DEC(t)$
– $(nf(nbt(s, 11)), bot(s, 11)) = (r(l), ap(l, l)) \in DEC(s)$

Example 13. Let $t = s$: $r(ap(l, l)) = ap(r(l), r(l))$
– $(nf(nbt(t, 11)), bot(t, 11)) = (r(l), r(l)) \in DEC(t)$
– $(top(s, 1), nf(ntp(s, 1))) = (r(l), r(l)) \in DEC(s)$

Example 14. Let $t = s$: $R(l, \varnothing) = r(l)$. With patterns, $nf(t[c.l']_1) = R(l', c.\varnothing)$ and $nf(s[c.l']_1) = ap(r(l'), c.\varnothing)$, then
– $(nf(nbt(nf(t[c.l']_1), 2)), bot(nf(t[c.l']_1), 2)) = (R(l', \varnothing), c.\varnothing)$ and
– $(top(nf(s[c.l']_1), 1), nf(ntp(nf(s[c.l']_1), 1))) = (r(l'), c.\varnothing)$.
Since $nf(R(\varnothing, \varnothing)) \equiv nf(r(\varnothing)) \equiv \varnothing$, we get that $t =_{ind(\mathcal{R})} s$.

Example 15. Let $t = s$: $s(x + m(x, e(x, y) + y)) = s(m(x, s(e(x, y)) + y))$
– $(top(t, 1221), nf(ntp(t, 1221))) = (s(m(x, e(x, y))), x + m(x, 0 + y)) \in DEC(t)$
– $(top(s, 1211), nf(ntp(s, 1211))) = (s(m(x, e(x, y))), m(x, s(0) + y)) \in DEC(s)$
We have now to verify $t' = s'$: $x + m(x, 0 + y)) = m(x, s(0) + y)$
– $(top(t', 22), nf(ntp(t', 22))) = (m(x, y), x) \in DEC(t')$
– $(top(s', 22), nf(ntp(s', 22))) = (m(x, y), x) \in DEC(s')$

Example 16. Let $t = s$: $\Sigma(x) = \Sigma I(x, 0)$, With patterns, $nf(t[s(y)]_1) = s(y) + \Sigma(y)$ and $nf(s[s(y)]_1) = \Sigma I(y, s(0 + y))$ then
– $(top(nf(t[s(y)]_1, 2), nf(ntp(nf(t[s(y)]_1, 2))) = (\Sigma(y), s(y))$ and
– $(nf(nbt(nf(s[s(y)]_1), 211)), bot(nf(s[s(y)]_1), 211)) = (\Sigma I(y, 0), s(0 + y))$.
Since $nf(\Sigma(0)) \equiv nf(\Sigma I(0)) \equiv 0$, we have to verify $s(y) =_{ind(\mathcal{R})} s(0 + y)$. Since $(s(y), 0)$ is a common partition of both sides, we get that $t =_{ind(\mathcal{R})} s$.

Example 17. Let $t = s$: $(x * x) * (x * x) = x * (x * (x * x))$. With patterns, $nf(t[s(y)]_{22}) = (x * x) * (x * y + x)$ and $nf(s[s(y)]_{222}) = x * (x * (x * (y + x)))$, then
– $(nf(nbt(nf(t[s(y)]_{22}), 222)), bot(nf(t[s(y)]_{22}), 222)) = ((x * x) * x, (x * x) * (x * y))$
– $(nf(nbt(nf(s[s(y)]_{222}), 2222)), bot(nf(s[s(y)]_{222}), 2222)) = (x * (x * x), x * (x * (x * y)))$.
Since $nf((x * x) * (x * 0)) \equiv nf(x * (x * (x * 0))) \equiv 0$, we have now to verify $t' = s'$: $(x * x) * x = x * (x * x)$.

With patterns, $nf(t'[s(y)]_2) = x * (x * y + x)$ and $nf(s[s(y)]_{22}) = x * (x * (y + x))$,
– $(nf(nbt(nf(t'[s(y)]_2), 22)), bot(nf(t'[s(y)]_2), 22)) = (x * x, (x * x) * y)$ and
– $(nf(nbt(nf(s'[s(y)]_{22}), 222)), bot(nf(s'[s(y)]_{22}), 222)) = (x * x, x * (x * y))$.
Since $nf((x * x) * 0) \equiv nf(x * (x * 0)) \equiv 0$, we get that $t =_{ind(\mathcal{R})} s$.

Example 18. Let $t = s$: $e(x, y + z) = e(x, y) * e(x, z)$. Since no partition or patterns be defined, we apply GEN rule. We get the equation $t' = s'$: $(e(x, y) * e(x, z)) * x = e(x, y) * (e(x, z) * x)$. With patterns,
– $nf(t'[s(x')]_2) = ((e(x, y) * e(x, z)) * x') + (e(x, y) * e(x, z))$ and
– $nf(s'[s(x')]_{22}) = e(x, y) * ((e(x, z) * x') + e(x, z))$
We then have $pat(t', 2, s(x')) = e(x, y) * e(x, z) = pat(s', 22, s(x'))$. Thus, since $nf((e(x, y) * e(x, z)) * 0) \equiv nf(e(x, y) * (e(x, z) * 0)) \equiv 0$, we get that $t =_{ind(\mathcal{R})} s$.

7 Conclusion

We have presented in this paper a new method to construct proofs that usually require mathematical induction with strong generalized hypotheses and additional lemmas. The method is simple and allows us to obtain very elegant and natural proofs. We know how to handle simple fragments of arithmetic and how to apply the method to proofs of properties of programs computing over lists.

The method has some limitations, however. The requirement on the monomorphic rewrite systems, while natural for primitive recursive definitions, must be generalized to handle general recursive definitions. Further, we do not treat multiple conjectures whose proofs use each other in a mutually recursive fashion. This will be given in an extended version of this paper. Finally, most nontrivial program proofs involve conditional reasoning: it may be possible to generalize this method in some simple cases. An implementation of the method in the NICE-system is underway.

References

[1] L. Bachmair, *Proof by consistency in equational theories*, Proceedings of Third IEEE LICS, 1988.

[2] A. Bouhoula and J.P. Jouannaud, *Automata-driven automated induction*, Information and Computation **169(1)** (2001), 1–22.

[3] A. Bouhoula, E. Kounalis, and M. Rusinowitch, *Automated mathematical induction*, Journal of Logic and Computation **5(5)** (1995), 631–668.

[4] R.S. Boyer and J.S. Moore, *A computational logic*, Academic Press, NY, 1979.

[5] F. Bronsard, U. Reddy, and R. Hasker, *Induction using term orders*, Journal of Automated Reasoning **16** (1996), 3–37.

[6] H. Comon, *Inductionless induction*, Handbook of Automated Reasoning (J. A. Robinson and A. Voronkov, eds.), vol. 1, Elsevier and MIT Press, 2001, pp. 913–962.

[7] N. Dershowitz, *Completion and its applications*, Actes du Séminaire d'Informatique Théorique (Paris), 1997.

[8] N. Dershowitz and J.P. Jouannaud, *Rewriting systems*, Handbook of Theoretical Computer Science (J. van Leeuwen, ed.), vol. B, North-Holland, 1990, pp. 243–320.

[9] A. Ireland and A. Bundy, *Using failure to guide inductive proof*, Journal of Automated Reasoning **16** (1996), 38–85.

[10] J.P. Jouannaud and E. Kounalis, *Automatic proofs by induction in theories without constructor*, Information and Computation **82(1)** (1989), 1–33.

[11] D. Kapur and M. Subramaniam, *Lemma discovery in automating induction*, Proceedings of Thirteenth Int. CADE, 1996.

[12] E. Kounalis, *How to check for the ground-reducibility property in term rewriting systems*, TCS **106 (1)** (1992), 87–117.

[13] E. Kounalis and P. Urso, *Generalization discovery for proofs by induction in conditional theories*, Proceedings of FLAIRS-99 (Orlando, FL), AAAI Press, 1999.

[14] T. Walsh, *A divergence critic for inductive proof*, Journal of Artificial Intelligence Research **4** (1996), 209–235.

[15] H. Zhang, D. Kapur, and M.S. Krishnamoorthy, *A mechanizable induction principle for equational specification*, Proceedings of Ninth Int. CADE, 1988.

Appendix – Long Proofs

The following lemma is essential for the proof of Lemma 1. The definitions 6, 7, and 8 respect the specification of this lemma.

Lemma 2. *Let $\mathcal{R}$ be a ground convergent and left-linear rewrite system, f be a defined function in $\mathcal{F}_{\mathcal{R}}$, y be a term, θ be a ground substitution, and $l \to r$ be a rule in $\mathcal{R}_f$, $nf(l[y]_p\theta) = nf(r[y]_q\theta)$ if it exists just one q s.t. $l/p = r/q$.*

Proof. Since $\mathcal{R}$ is ground convergent, $nf(l\theta) = nf(r\theta)$. Let $x = r/q = l/p$, and let σ be a substitution replacing the occurrences of x by y. By the properties of the substitution, $nf(l\sigma\theta) = nf(r\sigma\theta)$.

But, $nf(l[y]_p\theta) = nf(l\sigma\theta)$ and $nf(r[y]_q\theta) = nf(r\sigma\theta)$ if and only if the lone occurrence of x in l and r is at positions p and q. Since $\mathcal{R}$ is left-linear, and since there is just one q s.t. $r/q = x$, we get that $nf(l[y]_p\theta) = nf(r[y]_q\theta)$. □

Lemma 1. *Let $\mathcal{R}$ be a monomorphic rewrite system, let t be a term, and let p a position in P_f. For any ground substitution θ, and for any ground terms in $\mathcal{R}$-normal form A and B,*

1. *if $p = DP(\mathcal{R}, t(\varepsilon))$, $nf(t[A]_p\theta) = nf(t[\bot_{type(t)}]_p\theta) \otimes A$*
2. *if $p = UP(\mathcal{R}, t(\varepsilon))$, $nf(t[A]_p\theta) = A \otimes nf(t[\bot_{type(t)}]_p\theta)$*
3. *if $p = UCP(\mathcal{R}, t(\varepsilon))$, $nf(t[A \otimes B]_p\theta) = nf(t[A]_p\theta) \otimes nf(t[B]_p\theta)$*
4. *if $p = DCP(\mathcal{R}, t(\varepsilon))$, $nf(t[A \otimes B]_p\theta) = nf(t[B]_p\theta) \otimes nf(t[A]_p\theta)$*

Proof. Case 1. Downward Position. Let $t[A]_p\theta \to t_1 \to t_2 \to \ldots \to t_n$ be the rewriting sequence leading to $nf(t[A]_p\theta)$. By the Definition 6 of DP, for every rewriting step $i \leq n$ there exist a position q_i, which is a series of RA or DP positions, such that $t_i/q_i = A$.

Since t_n is a ground term in $\mathcal{R}$-normal form, t_n is only built over constructors. Thus, q_n is a series of RA positions. Then, by Definition 4 of $\otimes$, $nf(t[A]_p\theta) = t_n[A]_{q_n} = t_n[\bot]_{q_n} \otimes A$[7]. Further, by the Lemma 2, $nf(t[\bot]_p\theta) = nf(t_1[\bot]_{q_1}) = \ldots = nf(t_n[\bot]_{q_n})$. So, we get that $nf(t[A]_p\theta) = nf(t[\bot]_p\theta) \otimes A$. □

Case 2. Upward Position. By induction on the structure of the term A, since A is a ground term in $\mathcal{R}$-normal form, we have two cases (see Sect. 2):
- Either $A = \bot$, and by the definition of $\otimes$, $nf(t[\bot]_p\theta) = \bot \otimes nf(t[\bot]_p\theta)$.
- Or, $A = c[A']$, and by Definition 7 of UP,

$$\begin{aligned} nf(t[c[A']]_p\theta) &= nf(c[t[A']]_p\theta) \\ &= c[nf(t[A']\theta)] && \text{(since } c(\varepsilon) \text{ is a free constructor)} \\ &= c[A' \otimes nf(t[\bot]_p\theta)] && \text{(by induction hypothesis)} \\ &= c[A'] \otimes nf(t[\bot]_p\theta) && \text{(by Definition 4)} \end{aligned}$$

In both cases, we get that $nf(t[A]_p\theta) = A \otimes nf(t[\bot]_p\theta)$. □

[7] In the proofs, since we deal with only one sort, we note $\bot$ the constant of that sort.

Case 3. Up-contextual Position. By induction on the structure of the term A,
– $A = \bot$, and by the definition of $\otimes$, $nf(t[\bot \otimes B]_p\theta) = nf(t[B]_p\theta) = \bot \otimes nf(t[B]_p\theta)$. Further, by Definition 8 of UCP, $\bot$ is the normal form of $t[\bot]_p$.
– or $A = c[A']$, and $nf(t[c[A'] \otimes B]_p\theta)$

$$\begin{array}{ll}
= nf(t[c[A' \otimes B]]_p\theta) & \text{(by Definition 4)} \\
= nf(r[t[A' \otimes B]_p]_q\theta) \text{ with } q = DP(\mathcal{R}, r(\varepsilon)) & \text{(by the definition of } UCP) \\
= nf(r[\bot]_q\theta) \otimes nf(t[A' \otimes B]_p\theta) & \text{(by Case 1)} \\
= nf(r[\bot]_q\theta) \otimes nf(t[A']_p\theta) \otimes nf(t[B]_p\theta) & \text{(by induction hypothesis)} \\
= nf(r[t[A']_p]_q\theta) \otimes nf(t[B]_p\theta) & \text{(by Case 1)} \\
= nf(t[c[A']]_p\theta) \otimes nf(t[B]_p\theta) & \text{(by Lemma 2)}
\end{array}$$

In both cases, we get that $nf(t[A \otimes B]_p\theta) = nf(t[A]_p\theta) \otimes nf(t[B]_p\theta)$. □

Case 4. Down-contextual Position. By induction on the structure of the term A,
– $A = \bot$, and by the definition of $\otimes$, $nf(t[\bot \otimes B]_p\theta) = nf(t[B]_p\theta) = nf(t[B]_p\theta) \otimes \bot$ Further, by Definition 8 of DCP, $\bot$ is the normal form of $t[\bot]_p$.
– or $A = c[A']$, and $nf(t[c[A'] \otimes B]_p\theta)$

$$\begin{array}{ll}
= nf(t[c[A' \otimes B]]_p\theta) & \text{(by Definition 4)} \\
= nf(r[t[A' \otimes B]_p]_q\theta) \text{ with } q = UP(\mathcal{R}, r(\varepsilon)) & \text{(by the definition of } DCP) \\
= nf(t[A' \otimes B]_p\theta) \otimes nf(r[\bot]_q\theta) & \text{(by Case 2)} \\
= nf(t[B]_p\theta) \otimes nf(t[A']_p\theta) \otimes nf(r[\bot]_q\theta) & \text{(by induction hypothesis)} \\
= nf(t[B]_p\theta) \otimes nf(r[t[A']_p]_q\theta) & \text{(by Case 2)} \\
= nf(t[B]_p\theta) \otimes nf(t[c[A']]_p\theta) & \text{(by Lemma 2)}
\end{array}$$

In both cases, we get that $nf(t[A \otimes B]_p\theta) = nf(t[B]_p\theta) \otimes nf(t[A]_p\theta)$. □

Proposition 1. *Every element of $DEC(t)$ is a partition of t.*

Proof. Lets prove that if $p \leq TP(t)$, then for any substitution θ, $nf(t\theta) = nf(top(t,p)\theta) \otimes nf(ntp(t,p)\theta)$.

By induction on the top path p, by the Definition 9, we have a base-case (ε) and four induction case:

- Either $p = \varepsilon$, by Definition 9,
  $nf(t\theta) = nf(t\theta) \otimes \bot = nf(top(t,\varepsilon)\theta) \otimes nf(ntp(t,\varepsilon)\theta)$
- Or $p = qp'$ with a such that $t = t[a]_q$, by the substitution properties,
  $nf(t[a]_q\theta) = nf(t[nf(a\theta)]_q\theta)$ and by induction hypothesis: either $p' \leq TP(t)$, and
  $$nf(t[nf(a\theta)]_q\theta) = nf(t[nf(top(a,p')\theta) \otimes nf(ntp(a,p')\theta)]_q\theta)$$
  – with, either $q = RA(\mathcal{R}, t(\varepsilon))$, and by Definition 4,
    $nf(t[nf(a\theta)]_q\theta) = nf(t[nf(top(a,p')\theta)]_q\theta) \otimes nf(ntp(a,p')\theta)$

– or $q = UP(\mathcal{R}, t(\varepsilon))$, and by Lemma 1.2,
$$\begin{aligned} nf(t[nf(a\theta)]_q\theta) &= (nf(top(a,p')\theta) \otimes nf(ntp(a,p')\theta)) \otimes nf(t[\bot]_q\theta) \\ &= nf(top(a,p')\theta) \otimes nf(t[nf(ntp(a,p')\theta)]_q\theta) \end{aligned}$$
– or $q = UCP(\mathcal{R}, t(\varepsilon))$, and by Lemma 1.3,
$$nf(t[nf(a\theta)]_q\theta) = nf(t[nf(top(a,p')\theta)]_q\theta) \otimes nf(t[nf(ntp(a,p')\theta)]_q\theta)$$

Or $p' \leq DP(t)$, and

$$nf(t[nf(a\theta)]_q\theta) = nf(t[nf(nbt(a,p')\theta) \otimes nf(bot(a,p')\theta)]_q\theta)$$

– with $q = DCP(\mathcal{R}, t(\varepsilon))$, by Lemma 1.4,
$$nf(t[nf(a\theta)]_q\theta) = nf(t[nf(bot(a,p')\theta)]_q\theta) \otimes nf(t[nf(ntp(a,p')\theta)]_q\theta)$$

In each case, by the substitution properties and by the definition of top and ntp, $nf(t[a]_q\theta) = nf(top(t,qp')\theta) \otimes nf(ntp(t,qp')\theta)$.

The proof with a bottom path ($p \leq DP(t) \implies nf(t\theta) = nf(nbt(t,p)\theta) \otimes nf(bot(t,p)\theta)$) is symmetric. □

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399